Autonomous Networking

Daniele Bertagnoli

2022/2023

1 Sensor Networks

These networks are infrastructureless-based networks, and use sensors to sense the environment. Each sensor is composed by:

- **Battery**
- **CPU**
- **Operative System**
- **Sensor units (Hardware)**
- **Antenna**

These networks are used in several contexts such as "environmental controls", "structural monitoring" and "healtcare systems". A WSN should be available for the longest period of time possible and must scale well. We can distinguish between several partecipants in a WSN:

- **Data Source**, sensors that sense a phenomena.
- **Sink**, device which receives information from the sensors.
- **Actuator**, device that controls other devices basade on data.

We can deploy a WSN by:

- **Random deployment**, dropping the sensors in a random way over the area.
- **Regual deployment**, dropping the sensors by following a well defined pattern.

- **Mobile deployment**, use mobile sensors that can be moved passively from water and wind to cover an area.

To share information sensors use the wireless channel, so we must care about the signal quality that decreases with the incereasing of the distance. Moreover, the wireless signal is affected by the interferences that can create to much "noise" inside the signal.

2 MAC Protocols

MAC stands for *Medium Access Control* protocols, their aim is to control how the shared medium is used by the devices. These protocols define when a node can use the shared channel and for how long. We use MAC protocols must:

- **Reduce the energy waste**, by reducing the:
  - **Collisions**
  - **Overhearing problem**, nodes receive packets that were not destinated to them.
  - **Idle listening problem**
  - **Reduce the packet overhead**
- **Increase the system scalability**
- **Fairness among the nodes**
- **Low latency**
- **High throghput**

MAC protocols can use two different communtication patterns:

- **Broadcast**, typically used by the sink to transmit some information to the nodes. The packets are propagated into the network from all the nodes.
- **Convergecast**, typically used to collect the data, all or a group of nodes send the data to the sink.

2.1 MAC Protocols in WSN

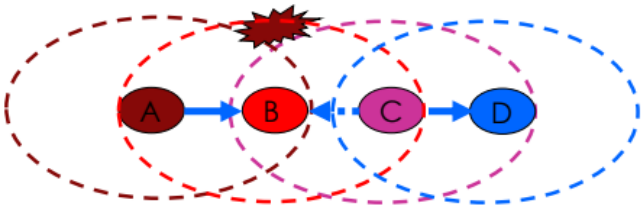
We can use two different types of MAC protocols in a WSN:

- **Contention based**, on-demand allocation for those that have packets to transmit. This is a scalable solution and does not require a central authority. We can have collisions, idle listening and interferences so an higher energy consumption. This protocols use **CSMA/CA** technique (*Carrier Sense Multiple Access Collision Avoidance*) in order to avoid the collisions (since the channel is shared we cannot detect them).
- **Scheduled based**, each node holds a schedule which defines when it can use the shared channel. So we cannot have collisions, but we need syncronization between the nodes and a central authority.

To avoid the hidden and exposed terminal problems, we use **RTS** (*Request To Send*) and **CTS** (*Clear To Send*) signals. The sender sends an RTS after the channel has been idle for a DIFS, the receiver waits for a SIFS and after that it will reply with a CTS. Collisions are still possible but only between RTS and CTS (that are less expensive to be retransmitted than a data packet).

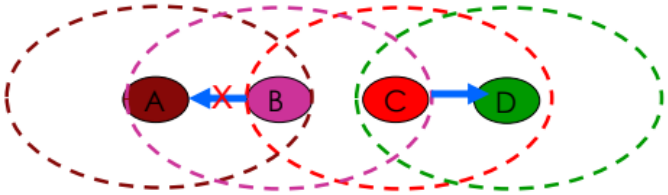
2.2 Hidden and Exposed Terminal Problems

2.2.1 Hiddeen Terminal Problem



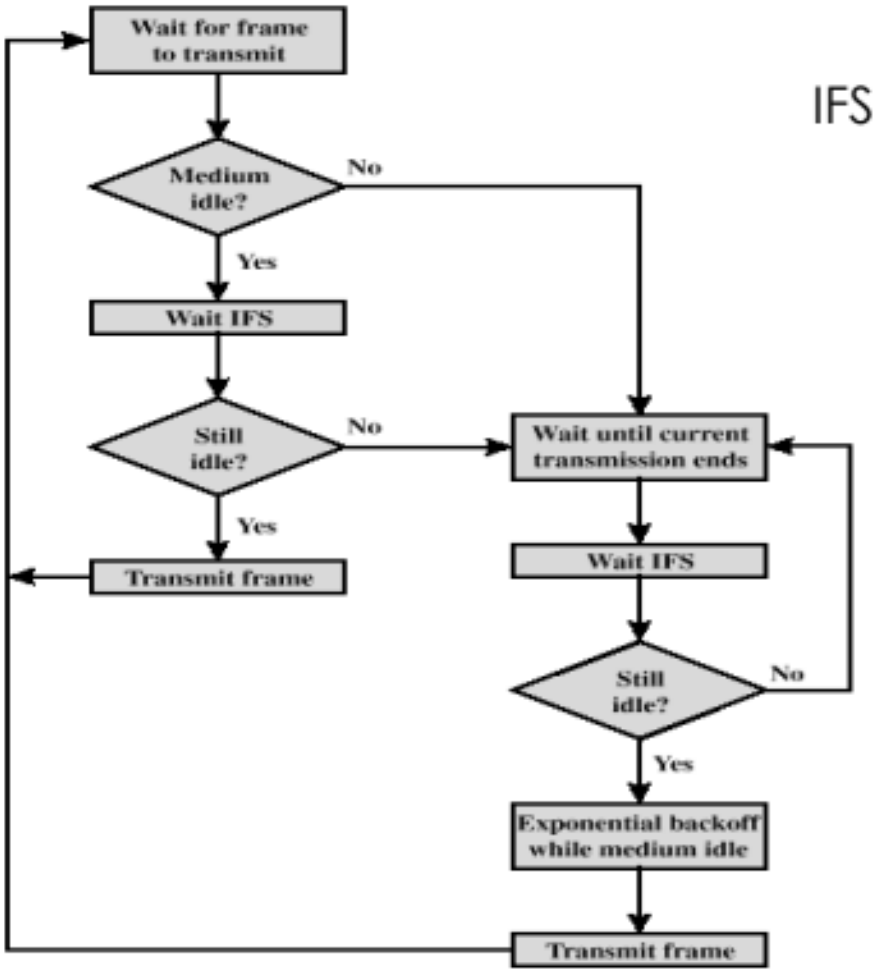
The node B can communicate with both A and C, but A cannot hear C and viceversa. When A transmits to B, C cannot detect the transmission so if C starts to communicate with D a collision will occur at B.

2.2.2 Exposed Terminal Problem



When C transmits to D, B detect the communication and does not communicate with A even if no collision will be generated in this case.

2.3 More deep in CSMA/CA



IFS: interframe spac

The *Exponential Backoff* is a random integer between [0,CW-1] where CW stands for *Contention Window*, a value that icreases with the increasing of the number of collisions. IFS means *Inter-Frame Space* and it's a fixed interval of time. We can define priorities with IFS:

- **SIFS (Short IFS)**, highest priority and it's used for ACKs, CTSs and control packets.
- **DIFS**, lowest priority used for the data.

2.4 S-MAC Protocols

S stands for "sleep", these protocols are used to avoid the idle listening problem. Each node implements a periodic sleeping and a periodic listening phase. This solution can reduce the energy waste but increses the latency. This protocol requires a syncronization between the nodes, since a node should be able to transmit a packet only when the destination is awake. Each node mantains a table with the neighbors' schedule, this table is periudically updated by using SYNC packets that are exchanged in the network. For the initial schedule, a node first will listen the channel for a certain amount of time, after that:

- if it has received a SYNC packet, it will become a **Follower** and will use the received schedule.
- if it has not received a SYNC packet, it will create its own schedule and send it to their neighbors becoming a **Syncronizer**.

In a large network we cannot guarantee a full syncronization between all the partecipants, so we can split the nodes into small groups in which each node follows the same schedule. The nodes that are between two groups will follow both of the two schedules. Since S-MAC is a contention based protocol, it will perform the carrier sense and uses RTS/CTS/DATA/ACK sequence.

2.5 NAV

Network Allocation Vector specifies how long the channel will remain busy, when a node receives CTS that was not destinated to him, the NAV is setted and the node will sleep for that period of time.

3 Routing Protocols

Routing is a technique used to share data between nodes and base stations through a multi-hop communication. Routing protocols must minimize the energy consumption and maximize the network lifetime.

We can distinguish between:

- **Flat Protocols**, all the nodes has the same importance
- **Hierarchical Protocols**, some nodes are most important than other.

We have three types of Flat Protocols:

- **Proactive Protocols**, know also as "Table driven protocols". Each node mantains a table (Destination Sequence Distance Vector DSDV) where it will store information about routes and reachable neighbors. When a path is needed, there is no extra delay due to the path discovery. These protocols require a lot of bandwidth and extra energy consumption to keep the tables updated.

- **Reactive Protocols**, operate on demand, routes are calculated only when needed. They differ for the technique used to discorver the new routes:

  - **Flooding**, each node forward the received packet to all of its neighbors (except the one by which the packet arrived). This technique creates a superflous amount of traffic and leads to a big energy waste, but we are sure that at least one packet will arrive to the destination by following the shortest path.
  - **Gossiping**, same ideas of flooding but each node choose randomly one of its neighbors for the packet forwarding. Less traffic generated but more slow than flooding.
  - **Dynamic Source Routing (DSR)**, each packet carries the entire followed path in its header. The sender sends a "route request" using the flooding technique. When one of the copies has arrived to the final destination, it will reply with a "route replu" which follows the stored path backward until it reaches the sender. In this way all the nodes inside the path can learn the path to reach the other nodes inside the same path. We have two main mechanisms:
    - **Route Discovery**
    - **Route Maintenance**
  - **Ad hoc On Demand Distance Vector routing (AODV)**, same idea of DSR for the discovery procedure. In this case, the nodes does not mantain the full path to reach a certain node, but it's computed on demand.

- **Geographic Routing**, each node holds a routing table which contains information about all the neighbors. These information are about the geographic position, the speed, the movement direction of that specific node. We can follow several strategies:

  - **Most forward within range r**, send the packet to the neighbor that realizes the most forward progress towards destination.
  - **Nearest node with any forward progress**, send the packet to the nearest neighbor (from us) that realizes a forward progress towards the destination.

4 RFID Systems

An RFID system is composed by:

- **RF Tags**, small devices without battery, or any compute capabilities. The can only reflect wirelss signals by using the backscattering technique.
- **Readers**, powerfull devices, their main goal is to query the tags.

These systems are typically used for the object localization (home devices, hospital devices, ...). The communtication can be:

- **Reader to Tag**
- **Tag to Reader**, tags cannot start a communtication, so in this case this is the reply to a reader's query.

Reader must care about communtication's collisions since the wireless channel is shared from all the tags. The reader can use several protocols to identify all the tags. We obtain a tag identification, when only one tag replies to a specific query.

4.1 MAC Protocols used in RFID systems

We can distinguish between two types of procols used for the tag identification:

- **Tree Based Protocols**, they uses tree to split the tags until we obtain a single reply:
  - **Binary Splitting**, the basic idea is to split the answering tag group into two divided subgroups until we get all leafs in the tree. Each tag has a counter, if this counter is equals to zero, then the tag can reply to the next reader's query. When the reader detects a collision (two or more replies), it notifies the replying tags, they will generate a random number and then this number it's added to their counter.
  - **Query Tree**, the reader makes query in a recursive way by using strings. A tag can reply if the received string is a substring of its ID. If the reader detects a collision, the string it's increased.
- **Aloha Protocols**, these protocols are based on time slotting. Slots are grouped in frames, a tag can use only one slot per frame to reply.
  - **Frame Slotted Aloha (FSA)**, uses the basic Aloha idea, when a tag has been identified, it will not partecipate to the next frames.

- **Tree Slotted Aloha (TSA)**, same idea of FSA, but when a collision has been detected, a new child frame is created. Only those tags that are responsable of the collision can partecipate to this new frame. We must estimate the tags population, if we underestimate it by choosing a small slot number for each frame, we may produce a big number of collisions.
- **Binary Splitting Tree Slotted Aloha (BSTSA)**, this protocol uses BS to split tags into groups whose size can be easily estimated. After that, TSA is used to perform the identification. By using BS, we can avoid the problem of estimating how many frames we should use in TSA to avoid collisions. At each tree level we can overestimate the number of tags in each group to create the frame.

To measure these protocols' performance we use:

- **System Efficency (SE)**: $\frac{number\ of\ identifications}{number\ of\ total\ queries}$
- **Time System Efficency (TSE)**: $\frac{number\ of\ identifications}{\beta * number\ of\ idle\ slots + number\ of\ identifications\ + number\ of\ collision\ slots}$

SE is used for Tree Based Protocols, the TSE is used for Aloha Protocols. We cannot use SE for Aloha Protocols since we must care about that an idle slot is less time consuming than a collision slot, so we must differenciate them. For both BS and QT the SE value is $\approx 38\%$. For FSA the TSE value is $\approx 60\%$, for TSA the TSE value is $\approx 70\%$.

5 Internet Of Things (IOT)

IOT indicates all the smart devices that are identificable and also are able to communicate and interact with the environment (some small compute capabilities). IOT key system-level features:

- **Scalability**
- **Ability to manage devices' hetereoginity**
- **Energy optimization**
- **Localization and computing capabilities**
- **Data management**
- **Self organization capabilities**
- **Security and privacy mechanisms**

5.1 SANET

SANET stands for **Wirelss Sensor/Actor Networks**, they are distributed Wirelss systems of hetereogeneous devices:

- **Sensors**
- **Actors**, collect sensors data and perform actions over the environment.

SANETs typically use RFID systems to locate and identify the devices. Through the **Backscattering technique** we can design smart devices that can work without batteries. Most of the sensors used in these networks are low energy consuming devices, so they can work with the backscattering technique. Two types of backscattering:

- **Ambient Backscattering**, devices harvest "power" from signals available in the environment (WiFi, Radio signal, ...). The main drawback is that the signal cannot be guaranteed and we have a low data rate.
- **RFID Backscattering**, devices harvest "power" from an RFID reader.

These devices have a **limited starage, limited range, limited power and low datarate**.

6 Unmanned Aereal Vehicol (UAV)

These devices are also know as **drones**, they are aircraft without a human pilot abroad. Theiy are able to sense large areas that cannot be accessed easily. Tipically multiple UAVs are deployed in a certain area, and they must be able to coordinate and cooperate to sense that area. Drones can act as **relay node** or **base station**. Each drone is equipped with several **radio modules**.

6.1 Comunication Protocols

Dronets are similar to WSN but with an higher mobility.

- Proactive Protocols, cannot be used because of their slow reactiveness to topology changements and low bandwidth.
- Reactive Protocols, can be a solution (routes are built on demand) but we need to consider the delay due to the route discovery.
- Hybrid Protocols, can be a solution (scale well with a low latency) but they are hard to implement.

- **Geographic Protocols**, commonly used solution. Each UAV uses its GPS to locate itself and its neighbors. Three main approaches:
  - **Greedy Forwarding**, we choose the geographically closest node to the target destination. The main drawback is the dead end problem.
  - **Store, Carry and Forward**, same greedy forwarding idea but to avoid dead end problem, the node will carry the packet until it will found a new node that is closer to the destination.
  - **Prediction Forwarding**, we choose the next node based on its direction, speed and distance respect to the destination.

7 Reinforcement Learning

RL is used to create an agent able to take a good sequence of decisions. In particular, RL will focus on the problem of learning while interacting whit an ever changing world (**Online Learning**). This technique is based on two main concepts:

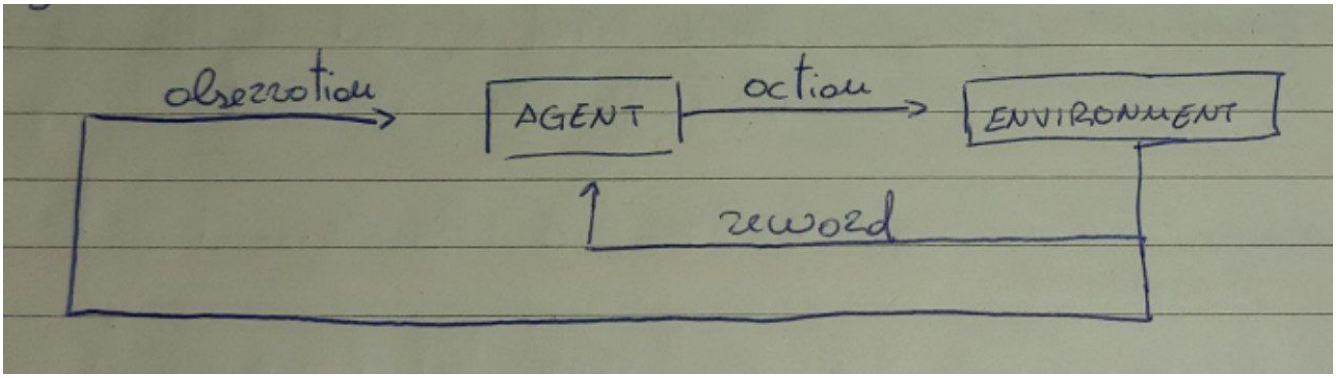
- **Trial and error research**, the agente must analyze its decisions and obtained rewards to take future actions.
- **Delayed rewards**, in many cases the taken action will not affect immediatly the real world or the gained rewards.

The agent must be able to sense the environment and perform actions in order to maximize the reward obtained. One of the RL's challenges is the tradeoff between:

- **Exploration**, the agent will exlpore new actions. So it takes some risks to collect information about unknown options.
- **Explotation**, the agent will take the best known action. So it takes advantage of the best option it knows.

We should guarantee a good balance between these two techniques. RL learns online, so while it is interacting with an evere changing world.

7.1 RL Framework



The agent observes the environment and then chooses an action to perform over it. After that, the real world gives a feedback (the reward) and the agent must analyze it to understand how good was the taken action. These are some important definitions:

- **Reward**, a reward R_i is a scalar signal which indicates how well the agent is doing ad the step i.
- **History**, is the sequence of observetions, actions and rewards.
- **State**, is a function used to determine what happen next: $S_t = f(H_t)$.
- **Environment state**, is the environment's private representation (usually non visible to the agent).
- **Agent state**, is the agent's internal representation (in a full observable environment the agent state is equal to the environment state). It can be any function of history. Our job is to build an agent state that is usefull in doing the best job in predicting what will happen next.
- **Policy**, is the agent's behaviour function, it maps states to actions. The function changes based on the obtained rewards. It can be:
  - **Deterministic**
  - **Stochastic**, based on probabilities.
- **Value function**, it specifies what is good in the long run (so it's a prediction about future rewards).

8 K-Armed Bandit Problem

We have a single state and we can choose among K actions, after each action we will receive a reward with a probability distribution. Our goal is to maximize the rewards.

$$Q_t(a) = E[R_t|A_t = a]$$

This is the **action value function**, it indicates the expected reward obtained by choosing the action a at timestep t. We can estimate this function by using two methods:

- **Sample avarage**, $Q_T(a) = \frac{sum\ of\ old\ rewards\ by\ choosing\ a}{number\ of\ times\ that\ a\ has\ been\ choosen}$
- **Incremental**:
  - For stationary problems, $Q_{n+1} = Q_n + \frac{1}{n}[R_n - Q_n]$, problems where the probability distribution does not change over the time.
  - For non-stationary problems, $Q_{n+1} = Q_n + \alpha[R_n - Q_n]$, problems where the probability distribution changes over the time.

After estimating the action value function, we need to choose the strategy to select an action:

- **Random**, we choose randomly among all the possible actions.
- **Greedy**, we choose the action with the highest Q-value.
- **ε-Greedy**, we choose the action with the highest Q-value with a probability $1-\varepsilon$, we choose a random action with a probability ε.
- **Optimistic initial value**, we assume that the initial Q-value is greater than zero. Then we apply a greedy or ε-greedy policy.
- **Optimism in the face of uncertainty**, typically used for stationary problems. We are choosing the action that possibly gives the highest reward. We use this formula to choose an action:

$$A_t = argmax_a[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}}] \text{ with:}$$

  - c = constant.
  - $N_t(a)$ = number of times that we took the action a.

9 Markov Decision Process

9.1 Definitions

A MDP is a classical formalization of sequential decision making problems, where the actions influence not just the immediate rewards but also the futere ones. We can build a **Markov Transition Matrix** which defines for each **Markov State** S the probability of transition to another Markov State S'. A state S is a Markov State iff it captures all the relevant information from the history.

$$MarkovTransitionMatrix = \begin{bmatrix} P_{1,1} & P_{1,2} & ... & P_{1,n} \\ ... & & & \\ ... & & & \\ ... & & & \\ P_{n,1} & P_{n,2} & ... & P_{n,n} \end{bmatrix}$$

A **Markov Reward Process** is a Markov chain with values. An MDP is a MRP with decisions. The **Return** G_t is the total discounted reward from timestep t:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

The **Value Function** $v(S)$ gives the long-term value of being in the state S, we can calculate through the Bellman equation:

$$v(S) = R_S + \gamma \sum_{S' \in \mathbf{S}} \mathbf{P}_{SS'} * v(S')$$

A **Markov Decision Process** is a Markov reward process with decisions. A **Policy** is a probability distribution over action given states:

$\pi(a|S) = P[A_t = a|S_t = S]$, is the probability of taking the action a starting from the state S.

Starting from the definition of policy, we can define a **Action Value Function** $Q_\pi(S, a)$ that gives the expected return starting from the state S and taking the action a by folliwing the policy π:

$$Q_\pi(S, a) = R_S^a + \gamma \sum_{S' \in \mathsf{S}} \mathsf{P}_{SS'}^a \; v_\pi(S')$$

Where $v_\pi(S)$ is the **Value Function** that follows the policy π, defined as follows:

$$v_\pi(S) = \sum_{a \in \mathsf{A}} \pi(a|S) \; Q_\pi(S, a)$$

9.2 Optimal Solution For MDP

To find the best solution for a MDP, we have to find the best policy that must be followed. First of all, we need to define:

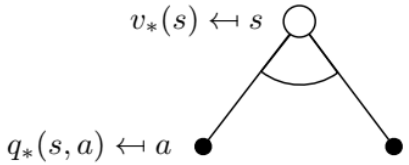
- $v_*(S) = \max_\pi v_\pi(S)$, so the best value function over all the policies.
- $Q_*(S, a) = \max_\pi Q_\pi(S, a)$, so the best action value function over all the policies.

Starting from there we can define a partial ordering over policies:

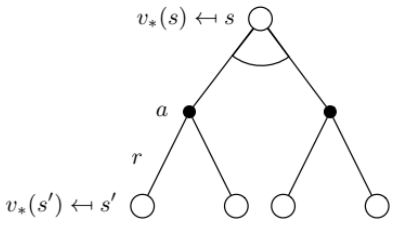
$$\pi \geq \pi' \text{ iff } v_\pi(S) \geq v_\pi(S) \ \forall S$$

For each MDP exists a policy π_* such that $\pi_* \geq \pi \ \forall \pi$. We can define two Bellman equations, one for the best value function and one for the best action value function:

1. $v_*(S) = \max_a Q_*(S, a)$

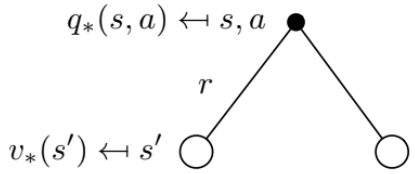


$$v_*(s) = \max_a q_*(s, a)$$

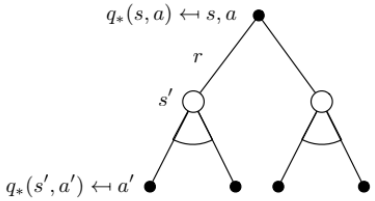


$$v_*(s) = \max_a \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s')$$

2. $Q_*(S, a) = R_S^a + \gamma \sum_{S' \in \mathsf{S}} \mathsf{P}_{SS'}^a \ v_*(S')$



$$q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s')$$



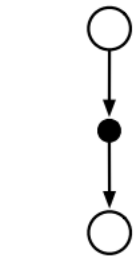
$$q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \max_{a'} q_*(s',a')$$

10 Temporal Difference Learning

TD methods learn directly from experience. It will estimate the value function v_π for a given policy π, so at each step the value function is updated. The **TD-Error** is how much should we adjust the v-value for the previous state. We can define the **TD(0)** method with the folliwing formula:

$$V(S_t) = V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$

$$\text{with TD-Error} = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$$



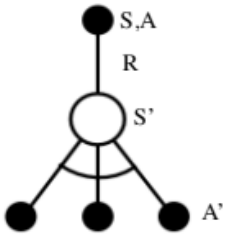
TD(0)

This method is called TD(0) because it's a special case of TD(λ).

11 Q-Learning

Q-Learning uses TD method for the **Control Problem**, find the optimal policy. The Bellman equation for Q-Learning is:

$$Q(S,A) = Q(S,A) + \alpha(R + \gamma \max_{a'} Q(S',a') - Q(S,A))$$



$$Q(S,A) \leftarrow Q(S,A) + \alpha \left(R + \gamma \max_{a'} Q(S',a') - Q(S,A) \right)$$

This $Q(S,A)$ converges to $Q_*(S,a)$.